"Express Mail" Mailing Label No. **EL436467572US**

PATENT APPLICATION
ATTORNEY DOCKET NO. NA99-08101

DETECTING COMPUTER VIRUSES OR MALICIOUS SOFTWARE BY PATCHING INSTRUCTIONS INTO AN EMULATOR

Inventor(s): Igor Muttik and Duncan V. Long

09586571.060100

BACKGROUND

Field of the Invention

The present invention relates to systems for detecting computer viruses and malicious software. More specifically, the present invention relates to a method and an apparatus for emulating computer viruses or other malicious software that operates by patching additional instructions into an emulator in order to aid in the process of detecting, decrypting or disinfecting code containing a computer virus or other malicious software.

Related Art

Malicious software, such as a computer virus, can enter a computer system in a number of ways. It can be introduced on a disk or a CD-ROM that is inserted into the computer system. It can also enter from a computer network, for example, within an email message.

If malicious software is executed by a computer system, it can cause a number of problems. The software can compromise security, by stealing passwords; by creating a "back door" into the computer system; or by otherwise accessing sensitive information. The software can also cause damage to the computer system, for example, by deleting files or by causing the computer system to fail.

Some types of malicious programs can be easily detected using simple detection techniques, such as scanning for a search string. However, this type of detection process can be easily subverted by converting a malicious algorithm into program code in different ways.

Another approach to detecting malicious software is to run a program on a real machine while attempting to intercept malicious actions. This technique, which is known as "behavior blocking," has a number of disadvantages. In spite of the attempt to intercept malicious actions, the program may nevertheless cause harm to the computer system. Furthermore, the behavior blocking mechanism typically cannot view an entire log of actions in making a blocking determination. Hence, the behavior blocking mechanism may make sub-optimal blocking decisions, which means harmless programs may be blocked or harmful programs may be allowed to execute.

Yet another approach to detecting malicious software is to "emulate" suspect code within an insulated environment in a computer system so that the computer system is protected from malicious actions of the suspect code.

One disadvantage to emulation is that it is almost impossible to provide complete emulation for all program instructions, all operating system calls and operating system environments that may be accessed by a piece of code being emulated without replicating the entire operating system in the process. Hence, in practice, emulators are typically able to emulate only commonly occurring program instructions and system calls.

This problem can be overcome by updating and recompiling an emulator to implement new system calls and new program instructions as different pieces of malicious software are encountered that make use of these new system calls and new program instructions. However, doing so can lead to logistical problems in keeping emulation programs up to date.

Another problem with current emulators is that they cannot deal with conflicting emulator environments. For example, one virus may be triggered by a system call returning the year 1999, while another virus is triggered by the same system call returning the year 2000.

What is needed is a method and an apparatus for emulating suspect code that can be easily reconfigured to accommodate new program instructions, system calls and emulation environments.

SUMMARY

One embodiment of the present invention provides a system for emulating computer viruses and/or malicious software that operates by patching additional program instructions into an emulator in order to aid in detecting a computer virus and/or malicious software within suspect code. During operation, the system loads a first emulator extension into the emulator. This first emulator extension includes program instructions that aid in the process of emulating the suspect code in order to detect a computer virus and/or malicious software. The system also loads the suspect code into an emulator buffer within a data space of a computer system. Next, the system performs an emulation using the first emulator extension and the suspect code. This emulation is performed within an insulated environment in the computer system so that the computer system is insulated from malicious actions of the suspect code. During this emulation, the system determines whether the suspect code is likely to exhibit malicious behavior.

In one embodiment of the present invention, loading the first emulator extension into the emulator involves loading the first emulator extension into the emulator buffer within the emulator. In this embodiment, performing the emulation involves emulating the program instructions that comprise the first emulator extension.

In one embodiment of the present invention, emulating the program instructions that comprise the first emulator extension causes the emulator to examine the suspect code looking for patterns that indicate that the suspect code is likely to exhibit malicious behavior.

In one embodiment of the present invention, emulating the program instructions that comprise the first emulator extension causes the program instructions within the first emulator extension to facilitate emulation of the suspect code.

In one embodiment of the present invention, prior to loading the first emulator extension into the emulator buffer, the system emulates the suspect code without using the first emulator extension.

In one embodiment of the present invention, the system additionally loads a second emulator extension into the emulator, and performs a second emulation using the second emulator extension and the suspect code. In a variation on this embodiment, the first emulator extension implements a first emulation environment that conflicts with a second emulation environment that is implemented by the second emulator extension.

In one embodiment of the present invention, loading the first emulator extension involves loading the first emulator extension from a database containing a plurality of different emulator extensions.

In one embodiment of the present invention, the first emulator extension includes code for decrypting an encrypted computer virus.

In one embodiment of the present invention, if a computer virus or other malicious software is detected within the suspect code, the system additionally disinfects the suspect code.

In one embodiment of the present invention, the first emulator extension facilitates emulating a non-standard computer instruction opcode.

In one embodiment of the present invention, the first emulator extension facilitates emulating an uncommonly used operating system call.

09585671.060100

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 illustrates a computer system in accordance with an embodiment of the present invention.

FIG. 2 illustrates the internal structure of an emulator for emulating and analyzing code for malicious behavior in accordance with an embodiment of the present invention.

FIG. 3 is a flow chart illustrating the process of emulating and analyzing code for malicious behavior using emulator extensions in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Computer System

FIG. 1 illustrates a computer system 106 in accordance with an embodiment of the present invention. Computer system 106 may include any type of computer system, including, but not limited to, a computer system based on a microprocessor, a mainframe computer, a digital signal processor, a personal organizer, a device controller, and a computational engine within an appliance.

Computer system 106 can receive suspect code 108 (which can potentially be malicious) from a number of different sources. Suspect code 108 may be introduced into computer system 106 by a remote host 101 across a network 102. For example, suspect code 108 may be included within an electronic mail (email) message from remote host 101 to computer system 106. Remote host 101 can include any entity that is capable of sending suspect code 108 across network 102 to computer system 106. Network 102 can include any type of wire or wireless communication channel capable of coupling together computing nodes. This includes, but is not limited to, a local area network, a wide area network, or a combination of networks. In one embodiment of the present invention, network 102 includes the Internet.

Suspect code 108 may additionally be introduced into computer system 106 by encoding suspect code 108 on a computer-readable storage medium, such as disk 104, and introducing disk 104 into computer system 106. Note that disk 104 can generally include any type of computer-readable storage medium, such as a magnetic disk, a magnetic tape and a CD-ROM.

Before executing suspect code 108, computer system 106 uses emulator 110 to analyze suspect code 108. Emulator 110 analyzes suspect code 108 by executing emulator code 203 and emulator extensions 204 as is described below with reference to FIGs. 2 and 3.

Emulator Structure

FIG. 2 illustrates the internal structure of an emulator 110 for emulating and analyzing suspect code 108 for malicious behavior in accordance with an embodiment of the present invention. Emulator 110 includes emulator code 203, emulator buffer 201 and database 206. Emulator code 203 includes code to perform the emulation.

Emulator buffer 201 is a protected region of memory (also known as a sandbox or a working space) in which suspect code 108 is stored and emulated. Emulator buffer 201 stores suspect code 108 as well as emulator extension 204. Emulator buffer 201 and emulator code 203 are designed so that while suspect code 108 that is executing within emulator buffer 201, suspect code 108 cannot damage or compromise computer system 106. Emulator extension 204 includes additional program instructions that assist emulator code 203 in the emulation process.

Note that emulator buffer 201 is not within the program space of computer system 106, but is instead in the data space. Hence, instructions within emulator extension 204 must themselves be emulated by emulator code 203. In an alternative embodiment of the present invention, emulator extension 204 is loaded as a patch into the program space of computer system 106. In this alternative embodiment, emulator extension can be executed directly on computer system 106.

Emulator extension 204 is retrieved from database 206, which contains a plurality of emulator extensions 208, which can be successively loaded into emulator buffer 201 during the emulation process. Database 206 can include any type of volatile or non-volatile memory or storage device that can be used to store emulator extensions 208. Database 206 can reside within computer system 106, or alternatively, can reside on an external database server that is separate from computer system 106.

During the emulation process, emulator extension 204 can read suspect code 108 looking for patterns indicating the suspect code 108 contains a virus or other type of malicious software. Alternatively, emulator extension 204 can set up an environment that is conducive to emulating suspect code 108. For example, emulator extension 204 can configure the system to emulate uncommonly used system calls or opcodes. This enables emulator code 203 and/or emulator extension 204 to determine of suspect code 108 exhibits malicious behavior. Emulator code 203 (working with emulator extension 204) ultimately outputs a decision 212 indicating whether suspect code 108 is malicious or not.

Note that emulator extension 204 can be emulated in a number of different ways. (1) Emulator extension 204 can be emulated as part of suspect code 108 by patching the emulator extension 204 into suspect code 108, possibly replacing, overlapping or overwriting portions of suspect code 108. In this case, the location where the patching occurs is defined in the database 206. (2) Emulator extension 204 can be executed before the suspect code 108 is executed, which enables emulator extension 204 to set up the environment that emulator extension 204 is responsible for handling. After this environment is set up, emulator extension 204 passes control suspect code 108. (3) Emulator extension 204 can replace suspect code 108 entirely. In this case, the suspect code 108 is not emulated at all, and emulator extension 204 produces decision 212 after analyzing the suspect code 108 as data. (4) Emulator extension 204 can be emulated after the suspect code 108 is emulated. This allows emulator extension 204 to analyze the results of running the suspect code 108 in order to produce decision 212.

Process of Emulation

FIG. 3 is a flow chart illustrating the process of emulating and analyzing code for malicious behavior using emulator extensions in accordance with an embodiment of the present invention. The system starts by receiving suspect code 108 from one of a number of possible sources as is described above with reference to FIG. 1 (step 302). The system loads this suspect code into emulator buffer 201 (step 304).

Next, the system runs emulator 110 (step 306). This causes suspect code 108 to be examined and/or emulated by emulator code 203. During the emulation process, the system determines whether or not suspect code 108 contains code that is likely to exhibit malicious behavior (step 308). If so, the system reports the malicious code to a system user or system administrator (step 310).

If no malicious code is detected, the system determines if there are any emulator extensions remaining in database 206 that have not already been used (step 312). If not, the system proceeds to the next file containing suspect code to repeat the entire process (step 314).

Otherwise, if there are emulator extensions remaining, the system loads the next emulator extension into emulator 110 (step 315). In one embodiment of the present invention, this involves loading emulator extension 204 into emulator buffer 201 within emulator 110. In an alternative embodiment, this involves loading emulator extension 204 into the program space of computer system 106 so that it can work in concert with emulator code 203 in performing a subsequent emulation.

Next, the system sets up emulator 110 to run emulator extension 204 (step 316). This may involve configuring emulator code 203 to initially run emulator extension 204. Next, the system returns to step 306 to continue with the emulation process using the new emulator extension.

Note that by using multiple emulator extensions it is possible to deal with conflicting emulator environments. For example, a first emulator extension can configure emulator 110 to detect a virus that is triggered by a system call returning the year 1999, while a second emulator extension can configure emulator 110 to detect a virus that is triggered by the same system call returning the year 2000.

The foregoing descriptions of embodiments of the invention have been presented for purposes of illustration and description only. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.